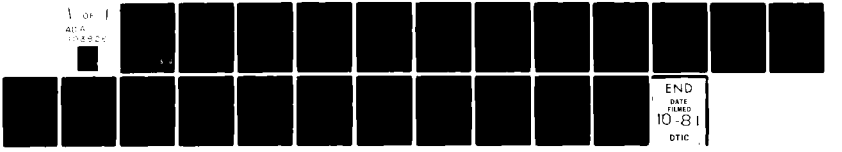
AD-A103 626 MASSACHUSETTS INST OF TECH CAMBRIDGE LAB FOR INFORMA--ETC F/G 9/2
QUERY PROCESSING IN DISTRIBUTED DATA BASES,(U)
JUL 81 V O LI
UNCLASSIFIED LIDS-P-1107 N00014-77-C-0532 NL



LEVEL II

(11) Jul[illegible]81

(12) (14) LIDS-P-1107

AD A103826

(6) # QUERY PROCESSING IN DISTRIBUTED DATA BASES*

by

(10) Victor O.K. Li

(12) 23

(15) *This research was carried out in part at the MIT Laboratory for Information and Decision Systems, Cambridge, MA, 02139 with support provided by the Office of Naval Research under contract ONR/N 00014-77-C-0532 (NR 041-519).

Paper submitted for publication to the IEEE Trans. on Software Engineering

DTIC FILE

Accession For	
NTIS GRA&I	X
DTIC TAB	☐
Unannounced	☐
Justification	
By Per Ltr. on file	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A	

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

DTIC ELECTE SEP 8 1981 S D

D

81 7 20 102

410950

QUERY PROCESSING IN DISTRIBUTED DATABASES

Victor O.K. Li
Department of Electrical Engineering
University of Southern California

Abstract

This paper describes two new distributed query processing algorithms. The MST Algorithm minimizes the total communication costs associated with a query while the MDT Algorithm minimizes the response time. These two algorithms are easy to analyze and to implement, since they are based on the minimum spanning tree and the shortest path problems, for which numerous algorithms exist. In addition, these two algorithms can be implemented using distributed computation, i.e., each node using only information available from adjacent nodes. We also develop the "artificial file node" technique to extend existing query processing algorithms which are designed for non-redundant databases to that for general, redundant databases. The two algorithms are illustrated by simple examples.

1. Introduction

A distributed database (DDB) consists of copies of datafiles (often redundant) distributed on a network of computers. Some enterprises, such as military Command, Control and Communications systems, are distributed in nature; since command posts and sensory gathering points are geographically dispersed, users are necessarily dispersed. Other potential users are airline reservation systems, and electronic funds transfer systems. A typical user is an enterprise which maintains operations at several geographically dispersed sites, and whose acti-

vities necessitate inter-site communication of data. The distribution of data in a network also offers advantages over the centralization of data at one computer. These advantages include: improved throughput via parallel processing, sharing of data and equipment, and modular expansion of data management capacity. In addition, when redundant data is maintained, one also achieves increased data reliability and improved response time (See [8], [9]).

Query processing (or data retrieval) is an important problem in distributed databases. Accessing data distributed in different computer sites necessitates the transmission of data over communication links. Since communication delay is substantial, the database management system must devise an efficient strategy to coordinate data processing at local computer sites and data transmission between sites. This problem is enhanced in a redundant database because which of the redundant copies to access becomes an important issue.

There are very few reports of work on distributed query processing. Wong proposed an algorithm that is being implemented in SDD-1 [10]. This algorithm starts with the simple, yet feasible, solution of moving every file directly to one node. It then attempts to replace this set of moves M by two sets of moves M_1 and M_2 that are to be performed sequentially with local processing in between. The criteria is that (i) the combined cost of M_1 and M_2 is less than that of M alone, and (ii) M_1 and M_2 are the least costly among all pairs satisfying (i). This technique is applied recursively to each set of moves until no improvement can be made. Note that Wong's algorithm is a "greedy" algorithm and will result

in a local optimum but not necessarily a global one. Wong assumes that (1) one is indifferent as to where the final result is produced, so long as it is produced at one site, (2) the communication cost as a function of data volume and communication links is known, and (3) the sizes of resulting files after local operations are known.

Hevner and Yao [6] proposed a simple algorithm for a special class of queries. The major assumptions are: (1) the result is to be produced at the site where the query originates, (2) the communication cost between any two nodes is defined as a linear function $C(X) = C_0 + C_1X$, where X is the amount of data transmitted, and C_0 and C_1 are constants, (3) the costs of local processing are negligible compared to the communication costs, and (4) when file i of size S_i is processed with file j, the resulting file has size S_iP_j, where P_j, the selectivity parameter of file j, is between 0 and 1. Assumption (2) of Hevner and Yao's algorithm implies that the topology of the communication network and the queueing effects will be ignored.

More recently, Chiu [1] devised a dynamic programming solution for certain queries called tree queries. However, the applicability of his method is extremely restricted. Note that all the algorithms described above address themselves only to non-redundant databases, i.e., only one copy of each file is maintained in the database.

In this paper, we describe two new query processing algorithms. The MST Algorithm minimizes the total communication costs associated with a query while the MDT Algorithm minimizes the response time. These algorithms are significantly different from existing query processing algorithms in that (1) they can be easily generalized to solve the query processing problem in a redundant database, and (2) they can be implemented using distributed computation. The following is an outline of this paper. In Section 2 we discuss the assumptions of our algorithms. The MST and MDT algorithms will be described in Sections 3 and 4 respectively. A distributed implementation of the algorithms is described in Section 5. In Section 6 we compare the different algorithms and we conclude in Section 7 with suggestions for further research.

2. Query Processing Model

The basic architecture of a DDB consists of computer sites arbitrarily connected via communication links. Each computer site contains a DDB Management system and portions (often redundant) of the database. Our query processing algorithms are based on the following assumptions:

(1) Data are logically viewed in the relational data model [3] and the unit of data distribution is a relation.

(2) The result is to be produced where the query originates, i.e., the requesting node.

(3) The costs of local operations are negligible compared to the communication costs.

(4) The communication cost as a function of the data volume and links is known.

(5) All files accessed by the query have the same size.

(6) The selectivity parameter of all files is one. Therefore, when two or more files are processed together, the resulting file has the same size as either of the original files.

Assumptions (4), (5) and (6) imply that the cost of a query processing strategy is a function of the communication links employed in the strategy, irrespective of the volume of traffic on these links.

3. The MST Algorithm

The MST Algorithm finds the optimal query processing strategy that minimizes the total communication costs. Consider a communication network with L channels, in which the average delay T_i experienced by a message at channel i is a differentiable function of the message flow per unit time λ_i at the channel. One possible form of the function is that derived by Kleinrock [7] for his communication network model, in which $T_i = 1/(\mu C_i - \lambda_i)$, where C_i is the capacity of channel i and $1/\mu$ is the average length of the messages. The average delay T for all messages in the network is given by $T = \sum_{i=1}^{L} \lambda_i T_i/\gamma$, where γ is the total external arrival rate of messages into the network. Hence, $\frac{\partial T}{\partial(\lambda_i/\mu)} = \frac{C_i}{\gamma(C_i-\lambda_i/\mu)^2}$ = the incremental delay for all messages in the network per unit increase in flow at channel i, provided the increase in flow is small compared to the existing traffic at the channel. If we let $\frac{\partial T}{\partial(\lambda_i/\mu)}$ be the communication cost on channel i the MST Algorithm will output a strategy that minimizes $\sum_{i=1}^{L} \frac{\partial T}{\partial(\lambda_i/\mu)}$. The cost of a strategy is proportional to $\sum_{i=1}^{L} \frac{\partial T}{\partial(\lambda_i/\mu)}$ because of unit file size. Therefore, the MST

strategy will minimize the incremental delay for all messages in the network due to a particular query. Obviously, other communication costs can be used.

There are two cases to consider: non-redundant files and redundant files.

3.1 Non-redundant Files

In this case, each file accessed by the query has only one copy maintained in the database.

Define a directed tree as a directed graph without a circuit, for which the outdegree of every node is unity: the outdegree of the root node being zero. Note that our definition of a directed tree is different from the usual definition (see, e.g. [2]) in that our directed links point towards the root node, rather than outwards from the root node.

We next describe the MST Algorithm for non-redundant files. There are two algorithms: (1) the MST1 Algorithm restricts all file processing at the node set N, where N is the set of nodes consisting of the result node R and the nodes where the file copies accessed by the query are located, (2) the MST2 Algorithm allows file processing at all nodes.

The restriction of the MST1 Algorithm is necessary in order for it to be optimal. If file processing can be performed at any node in the communication subnetwork, then in order to find the optimal strategy, we have to solve a Steiner

problem, which is a much harder optimization problem than the MST. This will be discussed in more detail in Section 3.2.

The MST1 Algorithm

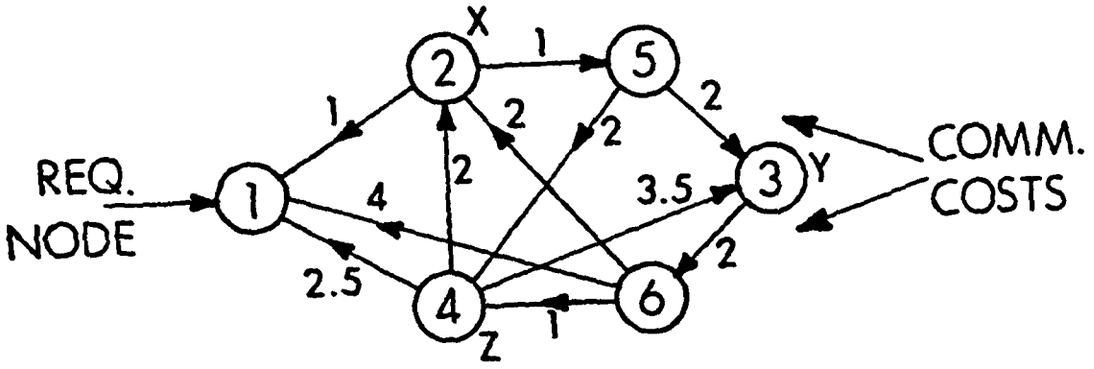
(1) Using the communication costs on the links of the communication subnetwork as input to a Shortest Path Algorithm, find the shortest paths between every pair of nodes in N. In general, the shortest path from node i to node j will have different length from the node j to node i path.

(2) Construct a directed graph G with node set N, and links with weights equal to the shortest path lengths between nodal pairs as calculated in step (1). Link (i, j) does not exist if the shortest path between nodes i and j is infinite.

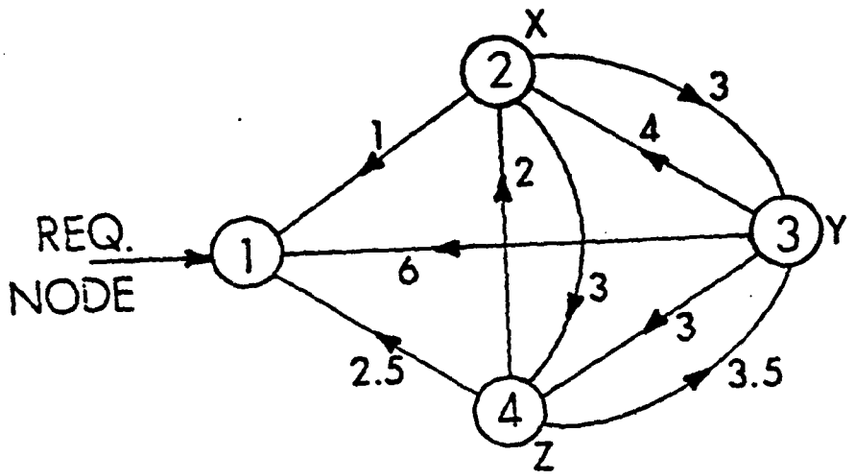
(3) Find the minimum weight directed spanning tree (MST) of G using node R as the root node of the tree.

(4) Each file is moved to the result node R using the directed path dictated by the MST. When two paths intersect, the two corresponding files are processed together, resulting in one file.

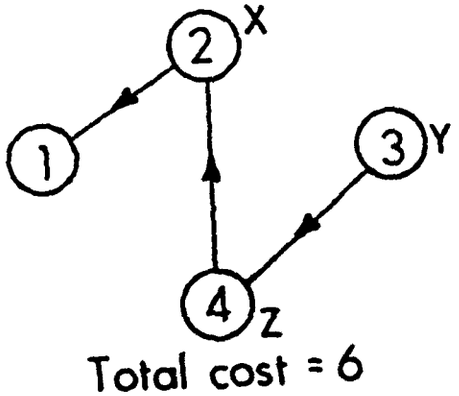
The algorithm is best illustrated by an example. Consider the six-node communication network shown in Fig. 1(a). A query originating at node 1 accesses files X,Y and Z at nodes 2, 3 and 4 respectively. The MST1 Algorithm says that we shall first find the shortest paths between every pair of



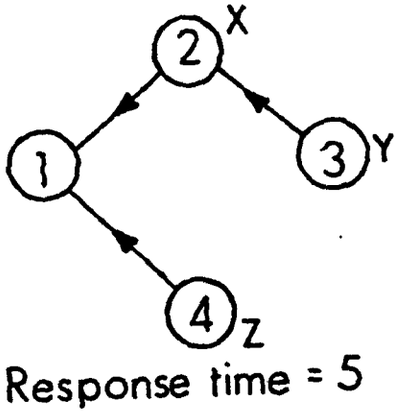
(a) Communication Subnetwork



(b) Graph G with Node Set N



(c). Minimum Spanning Tree(MST)



(d) Minimum Distance Tree(MDT)

Figure 1 Examples of the MST and MDT Query Processing Algorithms

nodes in the set of nodes {1,2,3,4}. We then construct a graph with node set {1,2,3,4} and links with weights equal to the shortest path lengths (see Fig. 1(b)). The MST consists of the directed links (3,4), (4,2) and (2,1) (see Fig. 1(c)). This means that file Y should be sent to node 4, to be processed with file Z. The resulting file is then sent to node 2, to be processed with file X, and the final result is then sent to node 1.

The correctness of the MSTl Algorithm is based on the following theorem:

Theorem 1 - Under the assumptions of the MSTl Algorithm, each query processing strategy is dominated by (i.e. more costly than) a spanning tree strategy with node R, the result node, as the root node. Proof: The specification that R is the root node is necessary since we want the result produced at R. Consider a directed graph G with node set N, i.e. the set of nodes consisting of R and the nodes where the files are located. Each query strategy corresponds to a subgraph of G. If the strategy does not correspond to a tree, some node $i \in N$ will have an outdegree greater than one, say two. This means that the file located at node i will be sent out twice through two different links. This is obviously inferior to just sending the file through one of the two links.

Q.E.D.

The MST1 Algorithm is optimal since the MST is the least costly among all spanning trees.

The MST2 Algorithm

(1) Using the communication costs on the communication links as the weights of the links, find a minimum weight tree that spans the node set N, with node R as the root of the tree.

(2) Each file is moved to the result node R using the directed path dictated by this minimum weight tree. When two paths intersect, the two corresponding files are processed together, resulting in one file.

Step (1) of the MST2 Algorithm corresponds to finding a solution to the Steiner problem, i.e. finding a minimum weight tree that spans a subset of the nodes of a graph.

3.2 Redundant Files

In this case, each file accessed by the query may have one or more copies maintained in the database.

By inventing artificial file nodes and artificial links of weight W connecting each file node with its copy locations, the MST Algorithm that we have developed for the non-redundant case can be easily extended to solve the redundant case.

Consider Fig. 2, where we have a request accessing two files X and Y. There are copies of X at nodes 2 and 3, and copies of Y at nodes 4 and 5. Note that we have created directed artificial arcs (X,2), (X,3), (Y,4) and (Y,5) with weights W. The direction of the artificial arcs for file X (or Y) ensures that only one of the arcs will be included in the optimal strategy. This is necessary since only one of the copies of X (or Y) will be accessed. The weight W can be chosen to be zero, or may be assigned to be different for different artificial links, to reflect the costs of accessing a file at different copy locations.

A strategy to satisfy a query originating at node 1 and accessing files X and Y will be represented by a tree spanning node 1 and the artificial nodes X and Y, but not necessarily spanning the whole node set. The optimal stragegy corresponds to the minimum weight tree of this type, i.e. a Steiner Tree.

In general, if node i is the requesting node and files X,Y,...Z are accessed in the query, then the optimal strategy corresponds to finding the minimum weight tree spanning the node set U, where U = {i,X,Y,...Z}.

The Steiner Problem can be solved by solving a number of MST's. For example, if we want to solve the Steiner Problem for the node set U in a graph with node set V, then we have to solve MST for all subgraphs of V that contain U (see [5]). Unfortunately, the number of subgraphs is large. However, because of the special structure of the query processing problem, only some of these subgraphs will

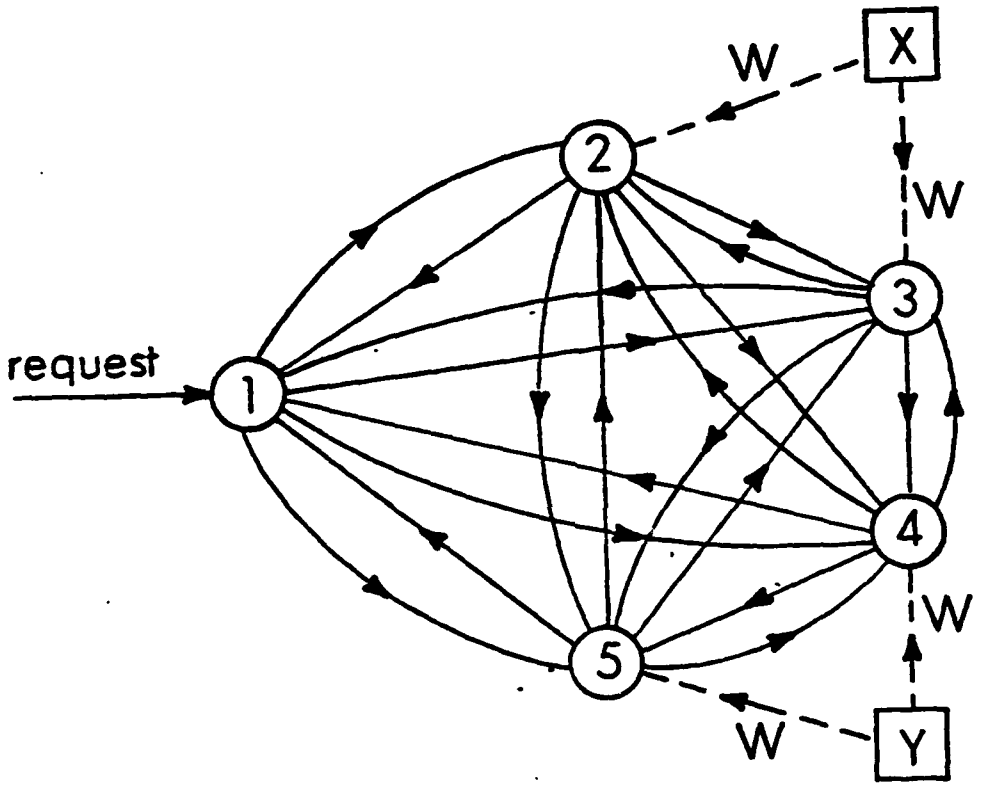


Figure 2 Query Processing with Redundant Files
-- The Steiner Algorithm

correspond to query processing strategies. Consider the example shown in Fig. 2; all we need to consider is the four subgraphs shown in Fig. 3.

In general, if we have n files and m copies of each file, the number of subgraphs to consider is m^n.

This artificial file node and artificial link technique can be used to generalize Wong's algorithm [10] to redundant databases. Hevner and Yao's algorithm [6] on the other hand, cannot be generalized easily, since it assumes that the communication costs between all pairs of nodes are the same and does not allow us to differentiate the costs of accessing different copies.

4. The Minimum Distance Tree Algorithm

The assumptions of the Minimum Distance Tree (MDT) Algorithm are the same as those of the MST Algorithm and are listed in Section 2. While the MST Algorithm minimizes the total communication costs, the MDT Algorithm minimizes the maximum of the communication costs for sending each file to the requesting node. In particular, if we designate the transmission delays on the communication channels as the communication costs, the MDT Algorithm will find the query processing strategy corresponding to minimum response time.

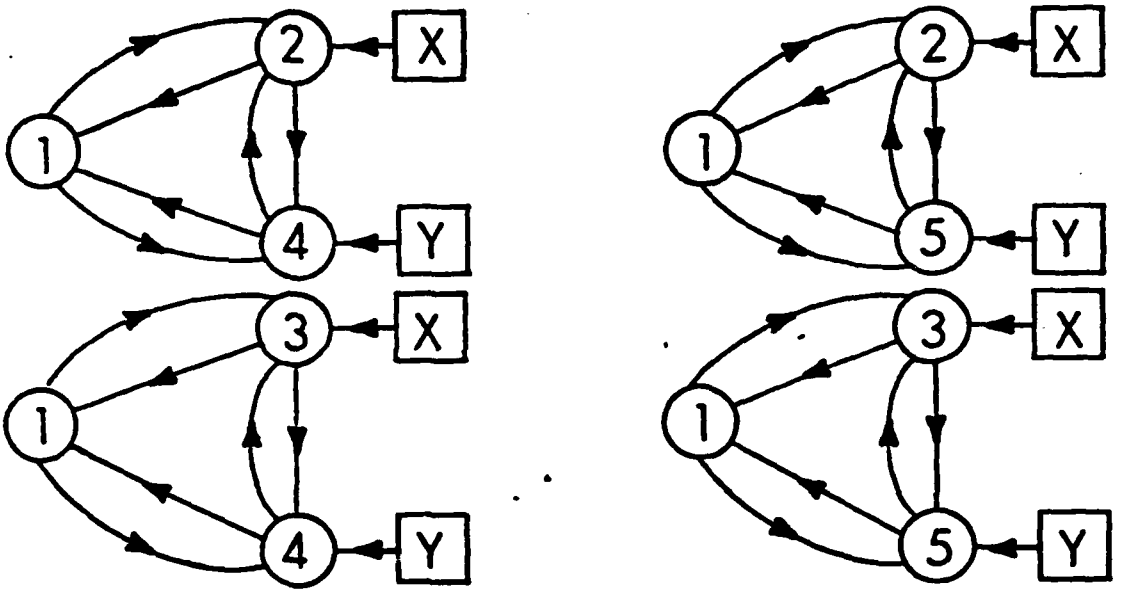


Figure 3 The Four Subgraphs to be Considered in the MST Solution of the Steiner Algorithm

The MDT Algorithm

(1) Construct a directed graph H of the communication subnetwork. The nodes of H are the nodes of the communication subnetwork and the links of H are the communication channels with weights equal to the communication costs of the channels.

(2) Create artificial file nodes for all files accessed by the query, and create artificial links connecting each file node and its copy locations. These artificial links should be directed outwards from the file nodes.

(3) Find the shortest directed paths from all file nodes to the requesting node. These shortest paths correspond to the paths taken by each individual file to reach the requesting node.

Note that the MDT Algorithm is the same for both redundant and non-redundant databases. In addition, since we are only interested in finding the shortest path for a file to reach the requesting node, the restriction that all file processing must be performed at the node set N can be removed. (Recall that N consists of the result node and the nodes containing copies of the files accessed by the query.)

An example of the MDT Algorithm is shown in Fig. 1(a), in which a query originating at node 1 accesses files X,Y and Z at nodes 2,3 and 4 respectively. The shortest paths from nodes 2,3, and 4 to node 1 are as shown in Fig. 1(d). These

are the paths taken by the files to reach the result node. The MDT Algorithm is based on the following theorem.

Theorem 2: Under the assumptions of the MDT Algorithm, it will minimize the response time. Proof: Consider a query accessing files 1,2,...n. The response time, by definition, is $\max_{1\leq i\leq n} F_i$, where F_i=time it takes a file i to reach the requesting node. Therefore, to minimize the response time, we have to min. $(\max_{1\leq i\leq n} F_i)$. The MDT Algorithm accomplishes this by minimizing each F_i, $1\leq i\leq n$.

Q.E.D.

In general, solutions that minimize response times are not unique. So long as the path taken by the file corresponding to the maximum communication delay is unchanged, the paths taken by the other files can be varied without affecting the response time.

5. Implementation of Algorithms

Previous research in query processing ([1], [6], [10]) assume that the query processing algorithms will be implemented in a centralized fashion. One of the computer sites, the central node, gets information from all other nodes on the delay on the communication links and uses this information to compute the optimal query processing strategy. This necessitates the transmission of information from all nodes in the network to the central node, plus the transmission of instructions

from the central node to other nodes to coordinate the query processing. There is also the problem of what to do when communication links and computers fail. The central node may not be able to get all the information it requires. In particular, if the central node fails, the whole system will fail. While the MST and MDT Algorithms can also be implemented using a centralized algorithm, they are significantly different from previous work in that they are particularly suited for distributed implementation, in which each node in the network bases all its decisions on information received only from its neighbors and it is not necessary to have a central node.

To implement the MST and the MDT Algorithms in a distributed fashion, we need a distributed computation version of the shortest path algorithm. Several such algorithms are described in Friedman [4]. These algorithms assume that each node in the network knows the communication delay to adjacent nodes. By communicating only with its adjacent nodes, each node can, by invoking one of these algorithms, then calculate the shortest paths from itself to every other node in the network. The computational complexity of these distributed algorithms at each node is similar to that for centralized implementation of the shortest path algorithms. The number of messages required is $O(Ln+n^2)$ where L, n are the number of links and the number of nodes in the communication network. Each algorithm can operate asynchronously, i.e. each node can start the algorithm and the algorithm remains correct even if any number of nodes independently decide to start the algorithm. Thus the shortest paths can be updated periodically to reflect changes in network topology and changes in communication costs on the links.

Once the shortest paths have been determined, implementation of the MST and MDT algorithms are relatively straight-forward. Finding the shortest paths corresponds to the pre-processing necessary for our algorithms. Suppose a query arrives at node R. It is assumed that each node has a directory of file locations so that node R can determine the set of nodes F it has to access to process this query.

To implement the MDT algorithm, node R sends messages to each node $i \in F$, requesting that the appropriate file at node i be sent to node R. Since each node i knows the shortest path to node R, no further coordination from R is necessary.

To implement the MST algorithm, node R has to find the MST, rooted at R, of a directed graph G, with nodeset $N = \{R\} \cup F$ and linkset with weights corresponding to the shortest path lengths between each pair of nodes in N. If some of these shortest path links are not already in memory at node R due to previous requests, node R must request them from the appropriate nodes. In the worst case, $2|F|$ messages will be necessary. Once the shortest path lengths are collected, the MST is easily found and instructions are sent to the file nodes to coordinate the processing of the query.

Note that under this distributed implementation, different nodes can initiate queries to the database independently and concurrently. In addition, so long as

the requesting node R and the file nodes F are in operation and communication paths exist between R and the file nodes, queries can be processed even when communication links and computers fail.

6. Comparison of Query Processing Algorithms

Wong's algorithm attempts to solve the most general query processing problem, where (1) files have different sizes, and selectivity parameters are arbitrary and (2) the communication subnetwork is completely general. Unfortunately, this general problem is very difficult and Wong's attempt only resulted in a heuristic giving local optimum.

Hevner and Yao looked at a simpler problem, relaxing requirement (2) by assuming that the communication cost between each pair of nodes depends linearly on the volume of data moved.

The MST Algorithm and the MDT Algorithm described in this paper relax requirement (1) by assuming same size files and selectivity parameters of value one. Our algorithms have the following advantages:

(1) We retain the traditional layering approach, i.e. separating the database from the underlying communication subnetwork. The latter will provide the input for our algorithms: the link costs in the MST and the MDT Algorithms.

(2) The minimum spanning tree problem and the shortest path problem in a directed graph are well understood and there exists efficient algorithms for their solutions.

(3) Under the assumptions we make, the MST and the MDT Algorithm will solve the problem optimally. Other query processing algorithms, for example, Wong's algorithm, only achieves a local optimum. In addition, Wong's algorithm only guarantees that the solution to the query will be available at one site, and is indifferent as to which site it is. The two algorithms proposed in this paper, on the other hand, guarantee that the result will be at the requesting node.

(4) The MST and MDT algorithms can be easily generalized to accommodate redundant file copies. While Wong's Algorithm can also be generalized using the same technique, Hevner and Yao's Algorithm cannot be generalized easily.

(5) The two algorithms proposed can be implemented using distributed computation.

7. Conclusions

Efficient query processing is an important function of distributed databases. In this paper we have developed two new query processing algorithms that are simple and can be implemented using distributed computation. In addition, we developed the artificial file node technique to handle redundant copies of data. These two

algorithms, however, suffer from very strict assumptions. In the future, we shall try to extend the algorithms by relaxing some of these assumptions. In addition, because of their assumptions, these two algorithms are actually hueristics, as is Wong's Algorithm [10]. It should be interesting to compare them.

REFERENCES

[1] W.D.M. Chiu, "Optimal Query Interpretation for Distributed Databases," Ph.D. Dissertation, Division of Applied Sciences, Harvard University, December 1979.

[2] N. Christofides, Graph Theory: An Algorithmic Approach, Academic Press, New York, 1975.

[3] C. Date, An Introduction to Database Systems, 2nd Ed., Addison-Wesley, 1977.

[4] D. Friedman, "Communication Complexity of Distributed Shortest Path Algorithms", Report LIDS-TH-886, MIT Lab. for Information and Decision Systems, Cambridge, February 1979.

[5] S.L. Hakimi, "Steiner's Problem in Graphs and Its Implications", Networks, 1, 1971, pp. 113-133.

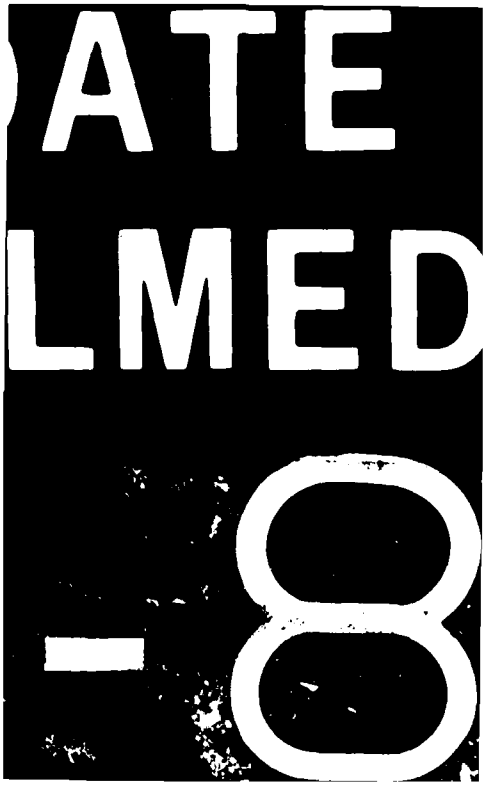
[6] A.R. Hevner and S.B. Yao, "Query Processing in Distributed Databases", IEEE Trans. on Software Eng., Vol. SE-5, No. 3, May 1979, pp. 177-187.

[7] L. Kleinrock, Queueing Systems, Vol 2, John Wiley & Sons, 1976.

[8] V. Li, Performance Models of Distributed Database Systems," Report LIDS-TH-1066, MIT Lab. for Information and Decision Systems, Cambridge, February 1981.

[9] J.B. Rothnie, P.A. Bernstein, S.A. Fox, N. Goodman, M.M. Hammer, T.A. Landers, C.L. Reeve, D.W. Shipman and E. Wong, "Introduction to a System for Distributed Databases," ACM Trans.on Database Systems, Vol. 5, No. 1, March 1980.

[10] E. Wong, "Retrieving Dispersed Data from SDD-1: A System for Distributed Databases," Rep. CCA-77-03, Computer Corp. of America, March 15, 1977.


ATE
LMED
8